

Count how many times a word appears in an array of words.

```
#region Copyright (c) 2004, Ryan Whitaker
```

```
/******  
*****
```

```
*****
```

```
,
```

```
' Copyright (c) 2004 Ryan Whitaker
```

```
,
```

```
' This software is provided 'as-is', without any express or  
implied warranty. In no
```

```
' event will the authors be held liable for any damages  
arising from the use of this
```

```
' software.
```

```
,
```

```
' Permission is granted to anyone to use this software for any  
purpose, including
```

```
' commercial applications, and to alter it and redistribute it  
freely, subject to the
```

```
' following restrictions:
```

```
,
```

```
' 1. The origin of this software must not be misrepresented;  
you must not claim that
```

```
' you wrote the original software. If you use this software in  
a product, an
```

```
' acknowledgment (see the following) in the product  
documentation is required.
```

```
,
```

```
' This product uses software written by the developers of  
NClassifier
```

```
' (http://nclassifier.sourceforge.net). NClassifier is a .NET  
port of the Nick
```

```
' Lothian's Java text classification engine, Classifier4J
```

```
' (http://classifier4j.sourceforge.net).
```

```

'
' 2. Altered source versions must be plainly marked as such,
and must not be
' misrepresented as being the original software.
'
' 3. This notice may not be removed or altered from any source
distribution.
'
'*****
*****/
#endregion

using System;
using System.Collections;
using System.Text.RegularExpressions;

namespace NClassifier
{
public class Utilities
{

///

/// Count how many times a word appears in an array of words.
///
/// The word to count. /// A non-null array of words. public
static int CountWords(string word, string[] words)
{
// find the index of one of the items in the array
int itemIndex = Array.BinarySearch(words, word);

// iterate backwards until we find the first match
if (itemIndex > 0)
while (itemIndex > 0 && words[itemIndex] == word)
itemIndex--;

// now itemIndex is one item before the start of the words
int count = 0;
while (itemIndex < words.Length && itemIndex >= 0)

```

```
{  
if (words[itemIndex] == word)  
count++;  
  
itemIndex++;  
  
if (itemIndex < words.Length) if (words[itemIndex] != word)  
break; } return count; } } } [/csharp]
```